

A Neural Network Approach for Predicting Microstructure Development in Cement

Dario Cruz¹, Douglas A. Talbert², William Eberle², Joe Biernacki¹

¹Chemical Engineering, Tennessee Tech University, Cookeville, TN, USA

²Computer Science, Tennessee Tech University, Cookeville, TN, USA

Abstract - Although portland cement concrete is the most widely used construction material in the world, efficient simulations for predicting hydration and property development (hardening and aging) are yet to be developed and made widely available to researchers and engineers. Current state-of-the-art simulations rely on computationally expensive models to generate spatially resolved time sequences of chemical and physical properties (microstructures) that allow engineers to predict permeability, strength, and durability. Improving simulation accuracy or efficiency would be a significant contribution. We describe a novel application of neural networks to simulate the microstructure of hydrating tricalcium silicate (the most abundant component of ordinary portland cement). Initial model predictions correlate well with benchmark models but require only a small fraction of the computation time.

Keywords: artificial neural networks, microstructure simulation, engineering application

1 Introduction

The properties of concrete, the most used material in the world by mass other than water, are largely due to portland cement, the binder that provides strength and durability. As such, proper understanding of cement hydration and property development (hardening and aging) in concrete is vital to infrastructure development and maintenance. Unfortunately, cement hydration is among science's enigmatic challenges, and ways to efficiently predict hydration and property development are widely sought [1].

Portland cement concrete is a complex composite consisting of aggregates (stones which provide volume, mass and stability to the concrete), cement paste (hydrated cement), unhydrated cement, and water-filled or partially water-filled pores. Furthermore, the aggregate, cement paste, and cement particles are themselves complex composites. Thus, the properties of concrete are largely dependent upon the proportions of these constituents and the chemistry and microscopic physical structure of the hydrated portland cement paste called *microstructure* (what the material physically looks like).

Predicting the progress of *hydration* and *microstructure* development is an ongoing challenge. Current modeling strategies require researchers to opt for either a

computationally intensive approach that provides detailed microstructure, in some cases requiring weeks of run time for massively parallelized code on supercomputing platforms [2], or a *computationally efficient approach* that sacrifices detail for computational speed [3]. The goal of *this work* was to develop and evaluate a novel *application* of machine learning to streamline computations associated with microstructure development and, thus, reduce the cost of predicting cement microstructure.

After presenting some background and related work, we describe our proposed approach followed by our experimental methodology, results, discussion, and conclusions.

2 Background and related work

Type I portland cement consists mainly of *alite*, a form of tricalcium silicate (C_3S). Hydration of alite has historically been described as a process characterized by five distinct stages defined by which chemical reactions control the physical and chemical changes (*kinetics*) in that stage [4]. Hydration starts with the formation of hydrate nuclei at various sites on the alite surface, a random or stochastic process. These nuclei grow over time, fill the pore space, cause the cement to harden and properties to change, e.g. strength to develop. Model-based prediction of these kinetic behaviors is thought to be a key step in understanding how to design and optimize cements and other constituents of portland cement concrete [5].

2.1 Existing models of concrete hydration

The details of the complex process of alite hydration are vigorously debated but have been implemented in a number of existing models, including HydratiCA [6], Mi-CBM [3], CEMHYD3D [7], and μic (pronounced *mīk*) [8]. We briefly present each model below.

Developed at the National Institute of Standards and Technology (NIST), HydratiCA uses a kinetic cellular automaton-based (or kinetic CA) approach [7][9] to capture the stochastic nature of microstructure development. The CA approach models the problem as a three-dimensional (3-D) array of particles with associated properties in a space defined by a collection of *voxels* (*i.e.*, a pixel with an assigned volume and location in 3-D space). Changes to the particles and their properties over time is determined by the repeated application of rules describing the interactions among neighboring

particles. HydratiCA was developed and optimized for the simulation of mineral hydration processes and is based on an asynchronous 3-D stochastic kinetic CA. CA models such as HydratiCA can be viewed as image-based models that produce detailed microstructures that can be viewed as simulated 3-D color pictures similar to those shown in Fig. 1.

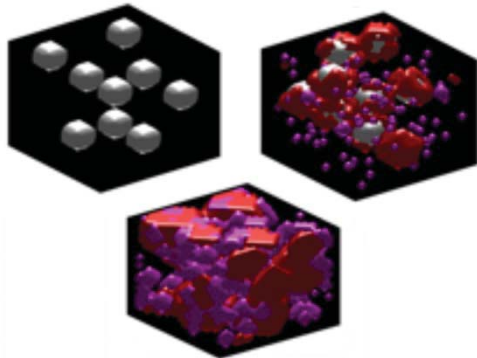


Fig. 1. Simulated Microstructure Images.

A predecessor of HydratiCA, CEMHYD3D simulates 3-D cement microstructures using carefully calibrated probabilistic rules [7]. In spite of some weaknesses, CEMHYD3D successfully generates microstructures that appear to reproduce a number of physical properties [10]-[12].

Developed by Bishnoi and Scrivener [8], μic is a user-definable modeling platform for representing microstructure and dynamics of microstructural change (kinetics). The authors define it as a tool-based “vector approach,” which stores information concerning the position, orientation and size of individual microstructural features as a vector. As is the case with the other microstructure-generating models, μic suffers from a high computational expense.

Mi-CBM (Multi-ionic Continuum-Based Model) is a continuum-based model that averages properties in predefined ways. This results in properties such as *average* density at a particular radius rather than density as a function of radius, azimuth and polar angle. While less computationally expensive than HydratiCA, Mi-CBM does not provide detailed microstructural information.

2.2 Machine learning in concrete and ceramics

In the field of cement hydrations, machine learning has only been used in select cases and ways. For instance, Devaney and Hagedorn [13], applied several machine learning techniques to discover new rules for classifying hydrating plaster over multiple time periods. They trained on X-ray micro-tomography data by using unsupervised classification, decision trees, and genetic programming to classify plaster powder and applied the learned rules to data at different hydration times. This work was also extrapolated to separate and distinguish un-hydrated plaster from gypsum crystals in the microstructure.

Wang and Yang [14], modeled early-age hydration kinetics of portland cement using flexible neural trees [15]

coupled with genetic algorithms to evolve the tree’s structure, rules, and parameters. The results show that the simulations and rate of hydration agreed well with experimental data.

However, neither of these approaches address the issue of microstructure development. *To the best of our knowledge, nobody has attempted to apply a machine learning approach for evolving cement microstructure.* One of the issues is that traditional machine learning classifiers are deterministic, and what we are dealing with in this domain are physical processes with some randomness. Some research, however, has dealt with predicting behavior in chaotic physical systems, such as weather forecasting [16].

3 Proposed autoregressive model (ARM)

Our proposed approach aims to improve the efficiency of microstructure generation by combining a CA with an artificial neural network. Specifically, we developed a set of non-linear autoregressive artificial neural networks [17], or *autoregressive model* (ARM), to predict the next state of alite (C_3S) particles given a set of prior states for the particles and their neighbors. Once the neural networks are trained, we initialize our model using a CA-based tool to generate an initial set of particle states, then feed those initial states into the neural networks to predict the next set of states. These predicted states are added to the CA-generated states and fed back into the neural networks to predict the *next* set of states. This cycle continues until states have been predicted for the desired number of time steps.

3.1 Cement hydration and CA state generation

For the CA phase of our approach, we developed an application we call *HydratiCA-lite*, based on stochastic rules for chemical reactions. HydratiCA-lite simulates the conversion of cement grains (C_3S) to cement hydrates (C-S-H(I), C-S-H(II) and CH). Nucleation sites (the location where the first hydrates form) are picked at random from among an available set of voxels and precipitating phases are allowed to grow (deposit). Currently, the morphology used for C_3S particles is spherical, but others can also be included.

To reduce the amount of time needed to simulate hydration, HydratiCA-lite was implemented as a parallel algorithm by dividing the full lattice of particles into smaller sub-lattices, each one belonging to a single *process*. This parallel HydratiCA-lite model effectively produces microstructure, local solution, and solid phase compositions. Global properties can also be computed by forming spatial averages over the system volume.

The chemical reactions shown in Fig. 2 for C_3S hydration along with their respective kinetic rate constants (k_j^i) were adopted from Bullard and Flatt [18]. These chemical reactions determine the nature of the microstructure formed; each chemical species contributes to the morphology and properties of it. The rate of formation and consumption of species is governed by the kinetic constants and associated rate laws. These play an important role in the kinetics of C_3S hydration. For more information, see reference [6].

3.2 Autoregressive model

The HydratiCA-lite output was used as input to the ARM and to generate the microstructure data to both train and validate the neural network. This HydratiCA-lite data represents the hydration and property development of cement as a function of time. Thus, this data is a series of microstructure snapshots at different times. The period of time between snapshots is defined as the *lag* [19].

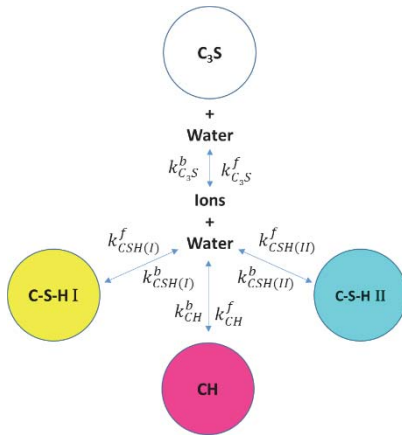


Fig. 2. Reaction Schema, Colors Map to Table I and Rate Constants to Tables II and III.

The neural network learns a function F that satisfies the autoregressive model defined by:

$$r(k) = F[r(k-1), \dots, r(k-n_r)] + e(k) \quad (1)$$

where the value of r at lag k depends on past values of r and the noise sequence $e(k)$ and where F is a function that produces the minimum error given a chosen objective (cost function).

Intuitively, $r(k)$ is the microstructure snapshot (at lag k) being predicted, and $r(k-1), \dots, r(k-n_r)$ are prior snapshots used to inform the current prediction. The use of multiple snapshots rather than only the one immediately prior to $r(k)$ enables the model to capture the rate of change (kinetics) of the reaction. The more general form of (1) includes an additional set of external information. In our case, however, such external information was not used.

To model (1), we use a Layered Digital Dynamic Network (LDDN) [20]. LDDNs support multiple recurrent (feedback) connections. Networks with such connections have been shown to be effective for time-series data modeling [17]. The structure of LDDNs also enable a type of memory that makes it possible to learn time-varying patterns (*i.e.*, their output at any given lag will depend not only on the current network input, but on the *history* or the *past* sequence of inputs).

It is precisely this feature of LDDNs that make them of interest for our task. The recurrent (autoregressive) network receives an input signal which will propagate forward through the network's layers, while *delayed* versions of the output (the network's results) and input signals, go into the input's layer through an arrange of delays that modifies the entering signals

(tapped delay lines) in order to successfully simulate time series behavior. Unlike *static* networks, the computed output of the network is a function of the current inputs, the weights and the biases of the network's layers, and the computed outputs from previous lags. More complete information regarding LDDNs can be found in [20].

3.3 Network training

The training *examples* were chosen to be three-dimensional (3-D) *Moore neighborhoods* of voxels. Thus, the number of voxels (3-D pixels) within a neighborhood is 27. Each of the neighborhoods contain unique voxels, thus the neighborhoods do not overlap with each other.

The software used for network simulation and training was MATLAB 2014a [21]. Training of the network used an internal mechanism to avoid over-fitting and to optimize the network outputs; this mechanism takes the data and distributes it randomly into three internal sub-categories: training, validation, and testing. The training data is used to learn network weights. The validation set is used to check for the stopping condition, and the testing set is used to evaluate different model parameters (*e.g.*, different weight arrangements).

The algorithm chosen for training uses a *Levenberg-Marquardt back-propagation* method. The cost function used during training is the *mean squared error (MSE)* of the network's outputs compared to the original target data [20].

The network is trained in two phases. First, we apply *open-loop* training that does not use the delayed feedback loops that enable "memory" in the LDDN network. Once that completes, the network starts with the final weights from the open-loop training phase and initiates the *closed-loop* training phase. This second phase makes use of the delayed feedback of the outputs in cost computations.

This dual approach seeks to enhance the performance of the neural network's recurrency by re-adjusting the weights according to feedback from previous outputs. This promises to reduce the effect of error propagation. The neural network is a standard two-layer feed-forward network having an input layer, one hidden layer and an output layer, with a sigmoidal transfer function in the hidden layer, and a linear transfer function in the output layer.

3.4 Data processing: color channel separation

Like HydratiCA, HydratiCA-lite's output can be viewed as simulated 3-D color images. To produce these simulated images, a specific color was chosen for each solid phase (*e.g.*, magenta= CH , yellow= $C-S-H(I)$, white= C_3S , cyan= $C-S-H(II)$). These choices were made so that each color channel (red, green, and blue) would convey different information about the chemical components in the concrete. Table I shows the mapping between chemical species (molecular entity) and color channels.

Table I. Species Assigned to Each Channel.

RED	GREEN	BLUE
C ₃ S	C ₃ S	C ₃ S
C-S-H (β)	C-S-H (β)	CH
CH	C-S-H (γ)	C-S-H (γ)

To fully leverage this information, we separate the output from HydratiCA-lite into three separate color channels. Thus, three different sets of inputs to the network were considered, each representing one color channel. Each channel varies continuously between 0 and 1, and each was trained and predicted separately [22], resulting in three different neural networks.

3.5 Data preprocessing: static voxel removal

Initial experiments showed that the presence of *static voxels* (voxels whose value does not change) during training significantly degraded performance of the model. To reduce the impact of such voxels, a threshold value was chosen for the number of dynamic voxels a neighborhood must have to be fed to the network. We arbitrarily chose a threshold value of nine. Thus, if a neighborhood contained fewer than 9 dynamic voxels, the neighborhood was discarded from the data. Unfortunately, this practice also discards some dynamic voxels (those in neighborhoods with fewer than 9 dynamic voxels) and results in incomplete predictions. Therefore these voxels are incorporated into a set named *excluded dynamic voxels* and are predicted separately.

4 Experimental methodology

4.1 Baseline simulations and data generation

For our experiments, we first ran HydratiCA-lite to generate data for 1200 simulated 3-D images of cement hydration under a predetermined set of reaction rates (kinetic constants). Each image represented successive 100-second intervals.

This data set served as the training data for our neural network. We then ran a second HydratiCA-lite simulation using the same set of kinetic constants. Due to the stochastic nature of the hydration process, the resulting data differed qualitatively and quantitatively (as expected and desired) from our training data. This second simulation served as test data for our experiments. These two data sets ($train_1$ and $test_1$) were used to test the time and accuracy of our proposed approach when the training kinetics match the test kinetics but nucleation sites differ.

We also wanted to evaluate the impact of varying the test kinetics. To do so, we ran a third simulation with HydratiCA-lite with a different set of kinetic constants and different nucleation sites. The resulting data ($test_2$) would be used to evaluate the ability of the trained model to adapt to different reaction rates.

Even though there is only one training set, the subscript on $train_1$ serves as a reminder that its kinetic constants match those of $test_1$. Each simulation produced 1200 3-D images. The kinetic constants used are shown in Tables II and III. All other simulation parameters were unchanged across the three simulations.

Table II. Kinetic Constants for $Train_1$ and $Test_1$

Reaction	k^f (mol/m ² s)	k^b (mol/m ² s)
C3S	5.20E-07	5.20E+10
C-S-H(I)	1.20E-06	3.97E+01
C-S-H(II)	1.00E-06	9.12E+06
CH	7.19E-06	1.14E+00
Ions	6.00E-02	9.96E-01

Table III. Kinetic Constants for $Test_2$

Reaction	k^f (mol/m ² s)	k^b (mol/m ² s)
C3S	5.20E-08	1.65E-06
C-S-H(I)	9.50E-11	6.60E+01
C-S-H(II)	8.00E-10	6.65E-03
CH	1.62E-07	3.79E-02
Ions	6.00E-02	9.96E-01

It is important to note the difference between the kinetic constants in Table II and Table III, and that even a small change in one of them can lead to a totally different behavior of the concentration of chemical species that conform the microstructure.

Table IV describes the results of applying the *color channel separation* and the *static voxel removal* (as described in Section 3.4) to the $train_1$ data set.

Table IV. Neighborhoods and Voxels per Color Channel

	RED	GREEN	BLUE
# of dynamic neighborhoods	203	489	614
Total # of static voxels	10508	8061	6578
Total # of dynamic voxels	1827	4401	5526

4.2 Experimental Simulations

For our experiments, we partitioned the data into sets of 20 simulated snapshots with 100 seconds between each snapshot. We call each 20 snapshot partition a *lag*. The training data consisted of 60 such lags. Since, in practice, our network will start with an initial set of lags generated by HydratiCA-lite, we trained each of the ARM's three (red, green, and blue) networks using only lags 30-60 via the two-phase training described in Section 3.3.

Evaluation was performed by seeding each network with the first 10 lags from the test data ($test_1$, then $test_2$). Then the

neural networks are used to predict the next lag. The data input window then slides one lag forward in time and the next lag is predicted. This repeats until all desired lags have been predicted.

In the following section, we evaluate:

- the correlation between HyradiCA-lite's simulated microstructure and the ARM's predicted microstructure,
- the comparison between HyradiCA-lite's simulated solution composition and the ARM's solution composition, and
- execution time

5 Results

Fig. 3 visually illustrates the similarities between the HyradiCA-lite-generated images and the neural network-generated images. The figure illustrates a series of 2-D cross-sections at various hydration times comparing the HyradiCA-lite outcomes for $test_1$ (Fig. 3a) to that for the trained model (Fig. 3b). These *snapshots* indicate good qualitative predictability of the ARM.

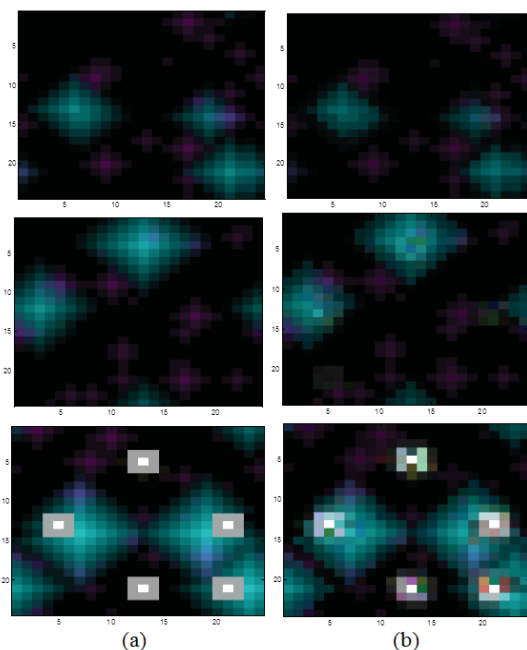


Fig. 3. Comparison of 2-D cross-sectional microstructures generated using HyradiCA-lite (a) and the neural network (b).

5.1 Microstructure correlation

Table V summarizes the correlation of the predicted voxels with the known test set voxels for each of the channels, indicating correlation between predicted microstructure and HyradiCA-lite generated microstructure.

Prediction accuracy is highly influenced by the presence of static voxels within the neighborhoods. Recall that all training neighborhoods must include at least nine dynamic voxels. Yet, some neighborhoods have fewer than nine dynamic voxels and are thus counted among the “excluded dynamic voxels.” Nonetheless, these voxels must be predicted

by the trained model in order to generate complete microstructures. This is error prone since the trained model does not have experience (opportunity) to learn from such neighborhoods.

Table V. R^2 Statistic for Last Step Predicted

SET	RED	GREEN	BLUE
Dynamic voxels ($Test_1$)	0.8053	0.8918	0.9191
Excluded dynamic voxels ($Test_1$)	0.7222	0.7319	0.7017
Dynamic voxels ($Test_2$)	0.8951	0.8346	0.79
Excluded dynamic voxels ($Test_2$)	0.7889	0.6754	0.789

5.2 Solution Composition

Global system profiles were generated and used to assess predictive accuracy of the trained model. Fig. 4 shows the global volume fractions of the solid phases for $test_1$ and $test_2$ compared to the network- (the trained model) predicted volume fractions. These results show that the prediction degrades immediately.

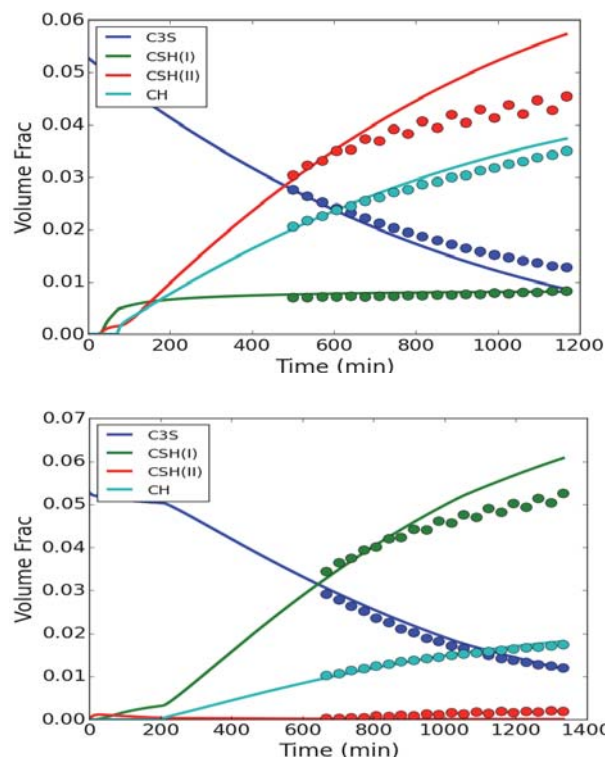


Fig. 4. Global outcomes for $test_1$ (a) and $test_2$ (b) for HyradiCA-lite (straight lines) and neural network (circles).

Regardless, the estimation error is not large and a correct evaluation and optimization of the prediction of the excluded dynamic voxels set would contribute to an enhancement of the

global prediction, as noted in Table V ($test_1$ results). The correlation between predicted values and the validation set values for the “excluded dynamic voxels” is considerably lower than for the dynamic voxels (an R^2 between 0.7 and 0.73 and between 0.8 and 0.92 respectively). Furthermore, the number of dynamic voxels also influences the prediction accuracy. In this case, the blue channel having the highest number of dynamic voxels exhibits the lowest error.

Unlike $test_1$, $test_2$ presented a larger amount of dynamic voxels in the red channel. Fig. 4b shows the global volume fractions computed by the HydratiCA-lite model and the dynamic neural network (predicted). In both cases depicted by Fig. 4a and 4b, the neural net predictions are in acceptable agreement with the results produced by the state of the art simulator (Hydratica-lite).

5.3 Execution Time

Table VI shows the CPU wall-clock time for HydratiCA-lite and the neural network, to predict microstructure for equivalent hydration times. The wall-clock time CPU speeds were measured on two different machines. HydratiCA-lite was run on an Intel Xeon 3.47 GHz multi-core processor using C++98. The autoregressive neural network was run on a Pentium 4 Xeon 2.8 GHz processor using MATLAB 2014a.

While this makes direct comparison of wall-clock times for the neural network and the CA models impossible, because MATLAB is deployed on a much slower machine but still clearly executes faster, it can be concluded that the neural network approach is computationally more efficient.

Table VI. Timing for 11.11 Hours of Hydration Simulation ($test_1$)

Simulator	CPU wall time for 11.11 hours of hydration (hours [s])	Number of processes
HydratiCA-lite	7.809 [28,112]	24
Autoregressive Neural Network	0.01556 [56]	1

6 Conclusions

These results show that the neural network (trained model) is able to make reasonable predictions for systems of unknown kinetics suggesting that there is considerable potential for the use of neural network models for fast generation of microstructure.

An autoregressive neural network with short memory was trained to generate microstructure for the hydration of tricalcium silicate (C_3S), the major component of Type I/II portland cements. This project utilized microstructures generated by a kinetic cellular automaton, a stochastic simulator. Global kinetic outcomes for the CA simulator were shown to be consistent with a continuum solution for identical kinetic and thermodynamic inputs. A single training data set was used for network learning.

The trained network was found to reproduce the training data set well, having an R^2 metric on the order of 0.98 for all

channels. The trained model did a good job at predicting microstructure for different initial conditions (different number and location of C_3S particles) producing R^2 values between 0.8 and 0.92, with the outcomes improving with the number of dynamic voxels to be predicted. The trained model did a surprisingly good job even when the kinetic parameters (rates of reaction) were drastically changed along with the initial conditions, resulting in R^2 values between 0.79 and 0.9. Again, the predictive accuracy improved with increasing numbers of dynamic voxels. The neural network was able to reduce computation time by a nominal factor of 500× even when run on a slower computing platform.

This novel demonstration of the use of autoregressive neural networks for the prediction of microstructure in hydrating cement paste suggests that the approach is a viable option for dramatically reducing the computation time of kinetic microstructure generators.

7 References

- [1] Thomas, J. J., Biernacki, J. J., Bullard, J. W., Bishnoi, S., Dolado, J. S., Scherer G. W. and Lutge, A., “Modeling and simulation of cement hydration kinetics and microstructure development,” *Cem. Concr. Res.*, 41(12), 1257-1278, 2011.
- [2] Bullard, J., Enjolras, J., George, E., Satterfield, S. and Terrill, J. “A parallel reaction-transport model applied to cement hydration and microstructure development,” *Modeling Simul. Mater. Sci. Eng.*, vol. 18, no. 2, pp. 1-16, 2010.
- [3] Biernacki, J. and Xie, T. “An advanced single particle model for C3S and alite hydration,” *J. Am. Ceram. Soc.*, vol. 94, no. 7, pp. 2037-2047, 2011.
- [4] Bye, G., Portland cement, London: Thomas Telford, 1999.
- [5] Biernacki, J., Bullard, J., Constantiner, D., Meininger, R., Juenger, M., Cheung, J., Hansen, W. and Hooton, R., “Paving the way for a more sustainable concrete infrastructure -a vision for developing a comprehensive description of cement hydration kinetics,” National Institute of Standards and Technology Special Publication 1138, 2013.
- [6] Bullard, J., “Modeling of Cement Paste Hydration and Microstructure Development,” NIST Information Technology Laboratory, 2010.
- [7] Bentz, D., “Three-Dimensional Computer Simulation of Portland Cement Hydration and Microstructure Development” *J. Am. Ceram Soc.*, vol. 80, no. 1, pp. 3-21, 1997.
- [8] Bishnoi S. and Scrivener K., “ μic : A new platform for modelling the hydration of cements”, *Cem. Conc. Res.*, vol. 39, no. 4, pp. 266-274, 2009.
- [9] Bullard, J., “A determination of hydration mechanisms for tricalcium silicate using a kinetic cellular automata model”, *J. Am. Ceram. Soc.* vol. 91, no. 7, pp. 2088-2097, 2008.
- [10] Garboczi, E. and Bentz, D., “Diffusivity-porosity relation for cement paste,” in *In Proceeding from*

seminar on sulfate attack mechanisms (pp. 259-64), Quebec, Canada, 1999.

- [11] Shane, J., Mason, T., Jennings, H., Garboczi, E. and Bentz, D., "Effect of the interfacial transition zone on the conductivity of Portland cement mortars," *Journal of the American Ceramic Society*, vol. 83, no. 5, pp. 1137-1144, 2000.
- [12] Lawson, J., Phan, L. and Davis, F., "Mechanical properties of high performance concrete after exposure to elevated temperatures," US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2000.
- [13] Devaney, J. and Hagedorn, J., "Discovery in Hydrating Plaster Using Multiple Machine Learning Methods," 2002. [Online]. Available: <http://www.nist.gov/itl/math/hpcvg/upload/discov2002.pdf>. [Accessed 02 May 2015]. [Garboczi and Bentz, 2000] Garboczi, E. and Bentz, D., "Percolation Aspects of Cement Paste and Concrete Properties and Durability," ACI Special Publication, 189, 2000.
- [14] Wang, L., Yang, B., Chen, Y., Zhao, X., Chang, J. and Wang, H., "Modeling early-age hydration kinetics of Portland cement using flexible neural tree," *Neural Comput & Applic*, vol. 21, no. 5, pp. 877-889, 2010.
- [15] Chen, Y., Yang, B., Dong, J. and Abraham, A., "Time-series forecasting using flexible neural tree model," *Information sciences*, vol. 174, no. 3, pp. 219-235, 2005.
- [16] Danforth, C. and Yorke, J., "Making Forecasts for Chaotic Physical Processes," *American Physical Society, Physical Review Letters*, April 14, 2006.
- [17] De Jesús, O. and Hagan, M., "Backpropagation Through Time for a General Class of Recurrent Network," International Joint Conf. on Neural Networks, Washington, DC, July 15-19, pp. 2638-2642, 2001.
- [18] Bullard, J. and Flatt, R., "New Insights into the Effect of Calcium Hydroxide Precipitation on the Kinetics of Tricalcium Silicate Hydration," *Journal of the American Ceramic Society*, vol. 93, no. 7, pp. 1894-1903, 2010
- [19] Perreault, E., Kirsch, R. and Acosta, A., "Multiple-input, multiple-output system identification for characterization of limb stiffness dynamics," *Biological Cybernetics*, vol. 80, no. 5, pp. 327-337, 1999.
- [20] Hagan, M., "Dynamic Networks," [Online]. Available: <http://hagan.ecen.ceat.okstate.edu/DynamicNetworks.pdf>. [Accessed 02 May 2015].
- [21] <http://www.mathworks.com/products/matlab/>
- [22] Verma, N., "Future image frame generation using Artificial Neural Network with selected features," in IEEE Applied Imagery Pattern Recognition Workshop (AIPR), 2012 IEEE , pp.1,8, 9-11, 2012.